Stacking for Misclassification Cost Performance

Mike Cameron-Jones and Andrew Charman-Williams

University of Tasmania, Launceston, Australia
Michael.CameronJones@utas.edu.au, A.CharmanWilliams@utas.edu.au

Abstract. This paper investigates the application of the multiple classifier technique known as "stacking" [23], to the task of classifier learning for misclassification cost performance, by straightforwardly adapting a technique successfully developed by Ting and Witten [19, 20] for the task of classifier learning for accuracy performance. Experiments are reported comparing the performance of the stacked classifier with that of its component classifiers, and of other proposed cost-sensitive multiple classifier methods – a variation of "bagging", and two "boosting" style methods. These experiments confirm that stacking is competitive with the other methods that have previously been proposed. Some further experiments examine the performance of stacking methods with different numbers of component classifiers, including the case of stacking a single classifier, and provide the first demonstration that stacking a single classifier can be beneficial for many data sets.

1 Introduction

Whilst the field of machine learning addresses a wide range of tasks, one of the most common is that of learning a classifier from flat attribute-value descriptions of items for which the class is known, with the aim of predicting well the classes of new items from their attribute values. The usual notion of predicting well is that of accuracy – making few mistakes.

However in many circumstances different forms of mistake are considered to be of different levels of importance, e.g. some systems of criminal justice are based upon a view that it is a greater mistake to punish the innocent than to fail to punish the guilty. In the case of many commercial classification-style decisions, the different costs of different forms of misclassification will be estimated and will influence the decision-making process. There is current interest in the topic of classifier learning for situations with misclassification costs, as evidenced at ICML 2000 in papers e.g. [10], and at the workshop on cost-sensitive learning organised by Margineantu, (notes available at http://www.cs.orst.edu/~margindr/Workshops/CSL-ICML2k/worknotes.html). Some work [22] has also considered other costs, such as the cost of measuring the values of attributes, e.g. in medical diagnosis it may not be considered worth conducting an expensive test in some circumstances. Here we consider only misclassification costs, hereafter referred to simply as costs, but unlike some authors e.g. [11], we do not restrict ourselves to the two class case.

Many different forms of classifier, and different forms of classifier learning method have been investigated in machine learning, and recently there has

been considerable interest in learning classifiers that combine the predictions of component classifiers. The two types of multiple classifier method that have been most commonly investigated are "bagging" (bootstrap aggregating) due to Breiman e.g. [4], and "boosting" developed by Freund and Schapire e.g. [12]. Both of these have been adapted to multi-class problems with costs, e.g. bagging (in work involving one of us) [5] and boosting style methods (hereafter simply referred to as boosting) by Ting and Zheng [21], and by Quinlan in C5, his commercial (http://www.rulequest.com) successor to C4.5 [16].

In this paper we look at an application of "stacking", a general idea due to Wolpert [23], successfully applied by Ting and Witten [19, 20], to the problem of learning classifiers for accuracy performance. The basic concept of stacking is that the classifier consists of levels of classifiers (or "generalisers"), with the outputs from each level being inputs to the level above. (Stacking can be seen to some extent as a generalisation of the sort of architecture typical in neural networks.) Ting and Witten developed a successful two level approach, with three classifiers of different types at the lower level. Their experiments showed that an important aspect of the success was producing probability-style estimates, not just class predictions, from the lower level, and their successful higher level generalisers also produced probability-style estimates. Good probability estimates enable good cost-based decisions, hence this paper follows up their work by adapting it to the cost context. Experiments are conducted to evaluate and compare the performance of some stacking variations against each other, their component classifiers, and other multiple classifier techniques for problems with costs.

The paper continues with a description of the stacking method used, then the experiments and results, and ends with conclusions and further work.

2 Stacking with Costs

When altering multiple classifier methods to make use of cost information one possible approach is to make the individual component classifiers take the costs into account, and then combine categorical predictions from these cost-sensitive classifiers, as was considered for bagging in [5]. However, here we take the approach of delaying the use of the cost information, using stacking to generate probability estimates from which the expected cost of each possible classification decision can be estimated, and the decision with least expected cost chosen. (Using p_i to stand for the estimated probability that an item is of class i, and c_{ij} to stand for the cost of predicting class j when the actual class of the item is i, the expected cost of predicting class j is estimated as $\sum_i p_i c_{ij}$). As most of the implementation for estimating the probabilities was intended to follow that of Ting and Witten, much of the description below follows theirs.

2.1 Learning Stacked Classifiers

As stated in the Introduction the stacked classifiers considered here consist of two levels. The lower, "level-0", contains individual classifiers that have been

learned from the training data as they would normally be learned for use as single classifiers, e.g. one of the classifiers is a decision tree produced by the application of Quinlan's C4.5 [16] to the training data. When the stacked classifier is used to classify an item, the level-0 classifiers are given the values of the attributes for the item, as they would be if they were going to be used as single classifiers, but produce class probability estimates rather than a categorical class prediction, hence the term "generaliser" is more appropriate than classifier. The stacked classifier's output is produced by the generaliser in the higher level, "level-1", which takes the outputs from all the lower level generalisers as its inputs, estimating the class probabilities from these.

While the process of learning the lower level generalisers is standard, the higher level has a less standard task in that while it predicts items' class probabilities, as the lower does, the "attributes" that describe items to the higher level are the probability estimates from the lower. Further, its training data should not be generated by simply using the lower level generalisers that were learned from all the training data, as the higher level generaliser would then be learning only from probability estimates made by generalisers that had seen the items for which they were making the estimates. Such estimates could be of very different accuracy to those produced by the lower level when classifying unseen items – the circumstance for which the higher level is being learned. Ting and Witten generated the training data for the higher level by 10-fold cross-validation (CV). The training data is divided into 10 equal (or near equal) parts. Then for each of these 10 parts, higher level training data is created by training the lower level generalisers on the other 9 parts, and producing their probability estimates for the items in this part. The 10 parts of higher level training data are then combined to learn the higher level generaliser.

2.2 The Lower Level Generalisers

The experiments reported here use three different types of lower level generaliser, as per Ting and Witten.

The instance-based approach (IB) used was a modification of the Waikato Environment for Knowledge Analysis (WEKA) reimplementation of Aha's IB1 [1]. (WEKA has been developed by Witten's group at Waikato, and a more recent version can be obtained from http://www.cs.waikato.ac.nz/~ml.) The WEKA implementation does distance weighted class probability estimation – the weight of each classifying instance is the reciprocal of its distance from the item to be classified, and the estimated probabilities are normalised to sum to 1. Following Ting and Witten, only the three nearest neighbours were used to estimate the class probabilities, and the distance measure was modified to use Cost and Salzberg's Modified Value Difference Metric [8] for discrete attributes.

The naive Bayesian (NB) approach used was that from WEKA, using Laplace style (m-) estimation of probabilities for discrete attributes, and normal distribution assumptions for continuous attributes. Again the normalised probability estimates were used.

The decision tree method used was a modification of C4.5 [16], which was modified as per [19, 20] to use a form of Laplace estimate for the class probability estimates at a leaf. Using M for the number of items of the most frequent class at this leaf, and N for the total number of items at the leaf, the probability estimated for the most frequent class is: $\frac{M+1}{N+2}$, and the probabilities of the other classes are proportional to their frequency, such that the sum of the class probabilities is 1.

2.3 The Higher Level Generalisers

Ting and Witten used two main types of higher level generaliser, but found that one, an instance-based approach was generally inferior, and our experiments on this method have shown it to be almost always worse then their other approach, and are not reported here. Their more successful type of higher level generaliser was a weighted sum method. At classification time this method estimates the probability of each class separately, as a weighted sum of the probability estimates from the lower level generalisers. Using p_{ij} for the probability estimated by lower level generaliser j for class i, the higher level generaliser estimates the probability of class k as $\sum_i \sum_j \alpha_{ijk} p_{ij}$, where the coefficients, α_{ijk}, are learned from the training data generated through CV. Ting and Witten experimented with some alternative methods for learning the coefficients, and here we use their suggestion of restricting the coefficients to be non-negative, and learning the coefficients separately for each predicted class minimising the sum of the squares of the training errors, using Lawson and Hanson's NNLS routine [13]. In addition to considering the general case in which the weighted sum estimating the probability for one class includes terms that are the lower level generalisers' probability estimates for other classes (as in [19]), we also report results here for the approach in which the weighted sum estimating the probability of a class only includes terms that are the lower level generalisers' probability estimates for the class being predicted (as in [20]). The latter will be referred to as the "no off-diagonals" case, as it corresponds to zeroing off-diagonal coefficients in a view of the relevant coefficient arrays.

The way in which the final probability estimate are used to make a misclassification cost based decision is such that the accuracy of the ratio of the estimates may be more important than the accuracy of their differences, so a higher level generaliser based upon minimising the log loss of probabilities may be preferable to one minimising the square loss. Hence, in addition to the previously used higher level generalisers, we have also used the parallel update logistic regression method of [7] which learns weighting coefficients to minimise training log loss. Similarly to the main form of the weighted sum method, the probabilities for each class are estimated independently, from all lower level class probability estimates. The logistic is applied to a weighted sum of inverse logistics of the lower level estimates. (The use of the inverse logistic was suggested by Rob Schapire subsequent to the original version of the paper.) Based on heuristic considerations, the coefficients were updated 30 $\times$ number of classes times.

3 Experiments

This section presents the results of the experiments on the stacking and other methods. The number of data sets publically available with true commercial misclassification costs is very small as such costs are usually commercially confidential. Hence our experiments here follow the example of some previous misclassification cost work, e.g. [21], in using a range of generated misclassification cost matrices for each data set, so as to enable use of a broad range of commonly accessible data sets.

The 16 data sets used for the experiments were chosen to have a variety of numbers of instances, classes, and discrete and continuous attributes, while favouring data sets that had been used before in previous multiple classifier and cost work. With one exception, the data sets were simply used as obtained either direct from the UCI repository [3], as generated from the programs at the UCI repository, or as obtained in the WEKA distribution. The exception was the "mushroom" data set in which a class stratified sample of one tenth (rounded up) of the full data set is used, as the full data set would pose an uninterestingly easy problem for the learning methods. The data sets will be recognisable by name to those in machine learning, but a few brief notes are needed to clarify which versions of some data sets are used: Heart Disease Cleveland (5 classes), Hypothyroid and Sick Euthyroid (3772 instances, 4 classes and 2 classes respectively), LED-24 (200 instances), Waveform-40 (300 instances).

All experimental results reported are average costs per instance, (for accuracy results from our earlier experiments see [6], and for alternative suggestions on what to report see e.g. [15] and Holte's paper in the ICML 2000 workshop, but note that some of the alternatives may only be practicable for two class problems). Each average cost reported is an average over ten experiments. Each experiment consists of randomly generating ten cost matrices and determining the performance of each learning method on each cost matrix using a ten-fold CV. The same splits of the data and cost matrices were used for all learning methods. (The outer CVs to assess the performance of learning methods are distinct from the inner CVs used in stacking, which is learning from the same training data as the other learning methods.)

The random cost matrices were generated in a manner similar to that of [21]. All the diagonal elements, which correspond to correct predictions, are given costs of 0, one other element is chosen at random to have cost 1, and the remaining elements are assigned (uniformly) random integer costs from 1 to 10 inclusive. (Cost matrices thus generated cannot be degenerate in the sense introduced by Margineantu at the ICML 2000 workshop.) We permit uniform cost matrices, which were deliberately excluded in [21] for the two class case – the only one in which they are likely to arise.

Table 1 shows the average misclassification costs per instance of each of the three types of lower level generaliser (IB, NB and DT), each making cost-sensitive predictions in accordance with its probability estimates, voting amongst the three cost-sensitive generalisers (Vote), averaging the probability estimates from the three lower level generalisers then making a cost-sensitive prediction (PSum),

Table 1. Cost results for lower level generalisers and some combined methods

Data set	IB	NB	DT	Vote	PSum	Cheap	Stack	PBag	C5	CB
abalone	1.634	1.606	1.600	1.447	1.408	1.505	1.344	1.341	1.588	1.359
colic	0.418	0.543	0.417	0.392	0.363	0.424	0.351	0.348	0.370	0.406
credit-a	0.383	0.640	0.341	0.330	0.330	0.347	0.323	0.279	0.293	0.296
credit-g	0.582	0.488	0.518	0.457	0.450	0.453	0.427	0.441	0.447	0.452
diabetes	0.529	0.483	0.495	0.430	0.423	0.479	0.415	0.409	0.442	0.432
heart	1.994	1.919	2.370	1.942	1.882	1.955	1.714	1.886	2.070	1.905
hypothyroid	0.491	0.234	0.021	0.193	0.162	0.021	0.020	0.018	0.018	0.020
led	1.587	1.643	1.901	1.543	1.504	1.636	1.636	1.565	2.018	1.616
mushroom	0.004	0.131	0.016	0.011	0.029	0.007	0.005	0.004	0.009	0.012
sick	0.064	0.130	0.029	0.038	0.051	0.029	0.027	0.020	0.019	0.025
sonar	0.280	0.913	0.689	0.423	0.324	0.280	0.290	0.388	0.317	0.492
soybean	0.260	0.514	0.384	0.265	0.267	0.261	0.226	0.320	0.309	0.435
splice	0.197	0.175	0.261	0.162	0.157	0.175	0.145	0.212	0.199	0.232
tumor	2.932	2.358	2.899	2.483	2.464	2.361	2.440	2.649		2.689
vowel	0.100	1.569	1.044	0.473	0.439	0.100	0.098	0.481	0.378	0.755
waveform	1.077	0.779	1.414	0.832	0.803	0.785	0.733	0.789	0.724	0.814
Rel	2.28	2.35	1.00	1.32	1.30	0.76	0.69	0.73	0.76	0.85
W/L	3/13	1/15	0/16	1/15	1/15	3/13	0/0	8/8	4/11	4/12

selecting the lowest cost of the three cost-sensitive lower level generalisers on the basis of a ten-fold CV on the training data (Cheap), cost-stacking using the three lower level generalisers with the weighted sum higher level generaliser using all lower level probability estimates (Stack), making cost-sensitive predictions on the basis of probabilities estimated by our implementation of Bauer and Kohavi's "p-bagging" [2] (using 30 rounds) with unpruned trees and "backfitting" as suggested (PBag), using C5 boosting (for 30 rounds) with cost-sensitivity (C5), and using our implementation of Ting and Zheng's Cost Boosting [21] (for 30 rounds) switching for uniform cost matrices to our implementation of the version of AdaBoost in [2] (using 30 rounds) as Cost Boosting does not boost on uniform cost matrices (CB). (The use of 30 classifiers in bagging and boosting is for approximately fair comparability of computational effort in training with stacking's 10-fold CV across 3 classifiers.) The results for C5 on the tumor data set are missing because the early version of C5 being used did not report misclassification cost results for data sets with so many classes. (Quinlan has confirmed that more recent versions do report such results.)

There is no universally accepted approach to the comparison across different data sets of the performance of different learning methods. Here we provide two forms of summary result. The first, Rel, is the average relative cost over the data sets, where a method's relative cost for a data set is its cost divided by that of the cost-sensitive decision tree method (DT) for the same data set. DT was chosen as the basis for this calculation because the main multiple classifier methods that we are interested in comparing are based on decision tree methods that are in some respect cost-sensitive. The other form of summary result, W/L,

is the method's "win/loss" performance relative to that of the Stack method, i.e. for how many data sets the reported results are better than those of Stack and for how many they are worse. This is reported as Stack is the main novel approach to the cost context that we are advocating. Both forms of summary result are calculated from figures less truncated than those shown in the tables. The summary figures for C5 exclude the tumor data set. (Stack's average relative cost performance excluding the tumor data set is 0.68.)

The results show that Stack has the best average relative cost performance of all the methods in table 1, and is ahead of all methods except p-bagging (with which it ties) in terms of the win/loss performance. Interpreting the win/loss performance in terms of the corresponding level of significance treating each performance as an individual one-tailed paired sign test, the results would all be significant at the 95% level, except for the comparison with C5 and with bagging. Thus Stack's summary performance is substantially better than that of all its constituent lower level generalisers, the simpler methods of combining them, and our implementation of Cost Boosting, better than that of C5, and similar to that of bagging. Given the reported general superiority of boosting over bagging in accuracy terms e.g. [17], the results here (and some smaller scale comparisons in [14]) raise the question as to whether the adaptations of boosting to the cost situation can be further improved, and additional experiments implementing a form of the two-class-only AdaCost [11] and Adaboost' [18] have shown more similar performance to bagging in the two class case.

As a referee has commented, it should be noted that a better average cost for a data set does not imply being better on every cost matrix tested, e.g. examining the results for the 4 data sets on which bagging and stacking perform most similarly, shows that out of the 400 random cost matrix draws, the method better on average is better on 227, tied on 19, and worse on 154. (In the 4 data sets in which they perform least similarly, the method better on average is better on 399 and tied on 1.)

Table 2 shows the average costs per instance for stacking each of the individual classifiers and using costs (IBSt, NBSt, DTSt), stacking pairs of classifiers (IBNBSt, IBDTSt, NBDTSt), stacking all three as before (Stack), stacking all three using weighted sums with no off-diagonals, (StNO), and stacking all three using the log loss based high level (StLog).

If the stacked individual classifiers are compared with their non-stacked versions, it can be seen that all three are better in terms of their average relative performance, and further all three stacked methods are better in terms of win/loss performance comparing directly against their non-stacked version, particularly the NB. Thus it seems that even stacking individual classifiers is generally beneficial, something which has not been previously demonstrated on such a variety of data sets. When the effect of increasing the number of stacked classifiers is examined, the average relative cost results show the desirable trend that each two classifier stacking technique has better performance than either of the two corresponding stacked individual classifiers, and the three classifier technique has better performance than any of the two classifier techniques. (Our

Table 2. Cost results for different stacking methods

Data set	IBSt	NBSt	DTSt	IBNBSt	IBDTSt	NBDTSt	Stack	StNO	StLog
abalone	1.389	1.395	1.401	1.353	1.362	1.362	1.344	1.398	1.345
colic	0.414	0.465	0.417	0.404	0.359	0.363	0.351	0.351	0.359
credit-a	0.390	0.463	0.350	0.363	0.330	0.303	0.323	0.321	0.321
credit-g	0.479	0.437	0.480	0.430	0.460	0.429	0.427	0.447	0.439
diabetes	0.477	0.415	0.448	0.412	0.435	0.409	0.415	0.420	0.422
heart	1.835	1.696	1.959	1.718	1.826	1.713	1.714	1.844	1.716
hypothyroid	0.372	0.231	0.021	0.230	0.021	0.020	0.020	0.020	0.021
led	1.618	1.672	1.958	1.572	1.668	1.669	1.636	1.572	1.558
mushroom	0.004	0.114	0.016	0.004	0.005	0.016	0.005	0.005	0.006
sick	0.065	0.091	0.030	0.062	0.028	0.028	0.027	0.027	0.028
sonar	0.290	0.468	0.465	0.301	0.278	0.442	0.290	0.288	0.274
soybean	0.252	0.415	0.393	0.238	0.235	0.286	0.226	0.228	0.221
splice	0.195	0.174	0.262	0.149	0.185	0.156	0.145	0.145	0.139
tumor	2.622	2.356	2.788	2.364	2.611	2.430	2.440	2.366	2.382
vowel	0.099	1.536	1.039	0.099	0.098	0.929	0.098	0.098	0.085
waveform	1.073	0.744	1.296	0.741	0.990	0.733	0.733	0.740	0.719
Rel	1.89	2.06	0.95	1.41	0.74	0.81	0.69	0.70	0.69
W/L	3/13	3/13	0/16	4/12	2/13	5/11	0/0	7/8	8/8

preliminary experiments in [6] also show that adding a fourth classifier, a neural network method, had potential.)

The comparison between the stacking methods shows that the use or non-use of the off-diagonals in the weighted sum approach is very evenly balanced consistent with Ting and Witten's results for accuracy performance. However, there is an aspect worth noting to the use or non-use of the off-diagonals, namely that the two data sets on which not using them seems noticeably beneficial are the LED and tumor data sets, which both have a low ratio of number of instances to number of off-diagonal coefficients, hence are cases where problems of overfitting are likely to be observed for the more complex model. Overall the log loss method performs similarly to the weighted sum.

4 Conclusions and Further Work

This paper has proposed the use in the misclassification cost context of some stacking methods for combining different types of underlying classifier, and experimentally compared (using 16 data sets) the cost performance of these methods against their constituent parts, against other methods of combining the same constituent parts, and against other cost-sensitive multiple classifiers that use bagging or boosting style methods. In these experiments the main stacking method's summary performance has been superior to that of nearly all the non-stacking alternatives considered and similar to that of the bagging approach. The issue of overfitting with the main method when there are too many coefficients relative to the amount of data has been raised for two data sets. The

possibility of using a log loss based higher level generaliser has been shown. The performance of bagged decision trees has been generally slightly superior to that of boosted decision trees, against the typical trend reported for performance in terms of accuracy. Experiments involving different numbers of constituent classifiers have shown that stacking can improve the performance of even a single classifier, and that performance generally improves with additional constituent classifiers.

The experimental results on stacking are generally encouraging, showing that the version considered here can be competitive with other previously proposed methods. By comparison with the bagging and boosting methods here, the stacking method has the benefit of using different types of constituent classifier, but the bagging and boosting methods can have the benefit of using more classifiers (of the same type) at classification time, e.g. 30 bagged decision trees versus one each of three types of classifier. Dietterich [9] has investigated the relative performance of bagging and boosting on decision trees in terms of the accuracy of the constituent classifiers produced and their diversity in terms of how much they differ in their mistakes, as such difference is essential to gaining a benefit from multiple classifier methods. It might be interesting to compare the diversity of different types of classifier with that of classifiers of the same type produced from boosting and bagging, to see how these compare.

While this paper has proposed and successfully demonstrated the potential of stacking in the misclassification cost context, there is much left to study, from the suggestion above involving investigating the current methods, through attempting to improve them, to the possibility of combining stacking with other methods to improve upon both.

Acknowledgements

This paper was completed while the first author was on study leave at AT&T Labs Research, whom the author wishes to thank, particularly Rob Schapire and Michael Collins. Thanks are also due to the UCI repository maintainers, and the contributors, e.g. R. Detrano for the Cleveland data, and M. Zwitter and M. Soklic for the primary tumour data which was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.

References

[1] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

[3] C. Blake, E. Keogh, and C.J. Merz. *UCI Repository of Machine Learning Databases*. University of California, Department of Information and Computer Science, Irvine, California, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
[5] M. Cameron-Jones and L. Richards. Repechage bootstrap aggregating for misclassification cost reduction. In *PRICAI'98: Topics in Artificial Intelligence – Fifth Pacific Rim International Conference on Artificial Intelligence*, pages 1–11. Springer Verlag, 1998.
[6] A. Charman-Williams. Cost-stacked classification, 1999. Honours thesis, School of Computing, University of Tasmania.
[7] M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 158–169. Morgan Kaufmann, 2000.
[8] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
[9] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40:139–157, 2000.
[10] C. Drummond and R.C. Holte. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 239–246. Morgan Kaufmann, 2000.
[11] W. Fan, S.J. Stolfo, J. Zhang, and P.K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Machine Learning: Proceedings of the Sixteenth International Conference (ICML '99)*, pages 97–105, 1999.
[12] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.
[13] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems*. SIAM, 1995.
[14] M.G. O'Meara. Investigations in cost boosting, 1998. Honours thesis, School of Computing, University of Tasmania.
[15] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98)*. Morgan Kaufmann, 1998.
[16] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. The Morgan Kaufmann Series in Machine Learning.
[17] J.R. Quinlan. Bagging, boosting and c4.5. In *Proceedings of the Thirteenth American Association for Artificial Intelligence National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.
[18] K.M. Ting. A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 983–990. Morgan Kaufmann, 2000.
[19] K.M. Ting and I.H. Witten. Stacked generalization: when does it work? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 866–871. Morgan Kaufmann, 1997.
[20] K.M. Ting and I.H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.
[21] K.M. Ting and Z. Zheng. Boosting trees for cost-sensitive classifications. In *Machine Learning: ECML-98: Proceedings of the Tenth European Conference on Machine Learning*, pages 190–195. Springer-Verlag, 1998.
[22] P.D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.
[23] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.